FVPP: Simboličko predstavljanje stanja BDD-ovima

Algoritmi u pozadini alata za provjeru modela

Pripremio: izv. prof. dr. sc. Alan Jović

Ak. god. 2022./2023.

Sadržaj

- Predstavljanje Booleovih funkcija
- Binarni dijagrami odlučivanja
- Algoritam ITE i implementacija BDD-ova
- Primjena BDD-ova

PREDSTAVLJANJE BOOLEOVIH FUNKCIJA

Simbolički postupci verifikacije sustava

- Rješenje problema eksplicitnih postupaka verifikacije provjerom modela:
  - Eksplozija stanja čini neefikasnim manipulaciju pojedinačnim stanjima u memoriji računala.
  - Potrebno je uvesti nove načine predstavljanja velikih skupova stanja u memoriji računala.
  - **Simbolički postupci** temelje se na uporabi **logičkih (Booleovih) funkcija**.
  - Simbolički postupci manipuliraju skupovima stanja a ne individualnim stanjima i time smanjuju potrebu za resursima.

Temeljne ideje u simboličkom pristupu

- Skupovi (relacije) kodiraju se Booleovim funkcijama.
- Booleove funkcije potrebno je prikazati i zatim njima upravljati u računalu na najučinkovitiji način (s aspekta prostornih i vremenskih zahtjeva).
- BDD dijagrami su vrlo učinkovita metoda predstavljanja Booleovih funkcija.
- Sve operacije nad skupovima stanja (izračun čvrste točke, dosezljivost, logičke operacije, itd.) obavljaju se s BDD dijagramima.

Predstavljanje skupova i relacija Booleovim funkcijama

- Skupove s konačnim brojem elemenata i binarne relacije nad tim skupovima moguće je prikazati Booleovim funkcijama i to najprije kodiranjem elemenata skupova odnosno relacija

- Neka je zadan skup stanja $\mathbf{S} = \{s_1, s_2, \dots , s_m\}$
- m elemenata skupa možemo binarno kodirati pomoću n bitova, $n = \lceil \log_2 (m) \rceil$ – najmanji cijeli broj veći od $\log_2 (m)$

- Binarnu relaciju $\mathbf{R} \subseteq \mathbf{S} \times \mathbf{T}$ također možemo kodirati.

Predstavljanje skupova i relacija Booleovim funkcijama

Primjer:

- S = {[0, 3], [8, 15], [16, 23]} – podskup skupa cijelih brojeva
- 20 elemenata, kodiranje s 5 binarnih varijabli: (x1, x2, x3, x4, x5)

Prvi element (0):	00000
Zadnji element (23):	10111

- Kodiranje (x1 teži 2^5, a x5 teži 2^0):

x1	x2	x3	x4	x5	f_S	elementi	broj elemenata (ukupno 20)
0	0	0	-	-	1	[0 - 3]	4 (x4, x5 sve kombinacije)
0	1	-	-	-	1	[8 - 15]	8 (x3, x4, x5 sve kombinacije)
1	0	-	-	-	1	[16 - 23]	8 (x3, x4, x5 sve kombinacije)

- **Karakteristična funkcija pripadnosti** elemenata skupa S: $f_S : \{0, 1\}^n \to \{0, 1\}$
  f_S = x1' x2' x3' + x1' x2 + x1 x2' (x3, x4, x5 u minimizaciji nestaju)
- 20 elemenata skupa S predstavljeno je sa samo 3 binarne varijable.
  Kodiranje se isplati !!!
- Tipično: broj binarnih varijabli kodiranja ≈ log(| S |)

Predstavljanje skupova i relacija Booleovim funkcijama

Primjer:

- $S = \{2, 4, 5, 7\}$
- $T = \{3, 5, 8\}$

1. Naći njihove karakteristične funkcije uz kodiranje:

- (x_i, y_k - težinske binarne varijable kodiranja)

s_i	x_1	x_2
2	0	0
4	0	1
5	1	0
7	1	1

t_i	y_1	y_2
3	0	0
5	0	1
8	1	0

- $\mathbf{f_S = 1}$ (!!!) $\mathbf{f_T} = y1'\,y2' + y1'\,y2 + y1\,y2' \mathbf{= y1' + y2'}$ (minimizirano)

Predstavljanje skupova i relacija Booleovim funkcijama

2. Naći karakterističnu funkciju relacije:

- $f_R = \{(s, t) \in (S \times T) \mid t = s + 1\}$

- Iz kartezijskog produkta izaberemo podskup parova koji zadovoljavaju zadanu relaciju. Slijedi podskup:
  $S = \{2, 4, 5, 7\}$
  $T = \{3, 5, 8\}$
  $R = \{(2, 3), (4, 5), (7, 8)\}$

- Uz ranije zadano kodiranje:

	x1	x2	y1	y2
(2, 3)	0	0	0	0
(4, 5)	0	1	0	1
(7, 8)	1	1	1	0

- Karakteristična funkcija relacije R je:

  $f_R = x1'\,x2'\,y1'\,y2' + x1'\,x2\,y1'\,y2 + x1\,x2\,y1\,y2'$

Predstavljanje Booleovih funkcija

- Kodiranje skupova i relacija binarnim varijablama traži **učinkovito predstavljanje logičkih (Booleovih) funkcija f(*x*) u računalu**

  $f(x) : B^{n} \rightarrow B,\ \ B = \{0, 1\}, x = \{x_1, x_2,\ldots, x_n\}$

- **Kako prikazati Booleovu funkciju u računalu? – SVI OVDJE NAVEDENI PRISTUPI SU NEUČINKOVITI:**

1. Predstavljanje logičkih funkcija **tablicom istinitosti**
2. Predstavljanje logičkih funkcija **mintermima i makstermima**
3. Predstavljanje logičkih funkcija u standardnom (dvorazinskom) obliku **sume produkata (SOP)** ili **produkta suma (POS)**
4. Predstavljanje logičkih funkcija u **nestandardnom** (višerazinskom) obliku
5. Predstavljanje logičkih funkcija **Booleovim kockama,** odnosno **Karnaughovim (K) tablicama**

Predstavljanje Booleovih funkcija

1. **Tablica istinitosti**
    - Prednost: kanonski prikaz (jedinstven, svaka varijabla se pojavljuje), razumljiv, jednostavna provjera ekvivalencije.
    - Nedostatak: eksponencijalni prostor za pohranu (za n varijabli 2^n redaka – interpretacija u tablici).

2. **Suma mintermi ili produkt makstermi**
    - Minterm (npr. $m_6 = x\ y\ z'$) ili maksterm (npr. $M_6 = x'+y'+z$) sadrže sve varijable, prirodno se dobivaju iz tablice istinitosti
    - Prednost: kanonski prikaz, razumljiv, jednostavna provjera ekvivalencije.
    - Nedostatak: nije minimalan oblik funkcije, a time se gubi na optimalnosti prikaza.

Predstavljanje Booleovih funkcija

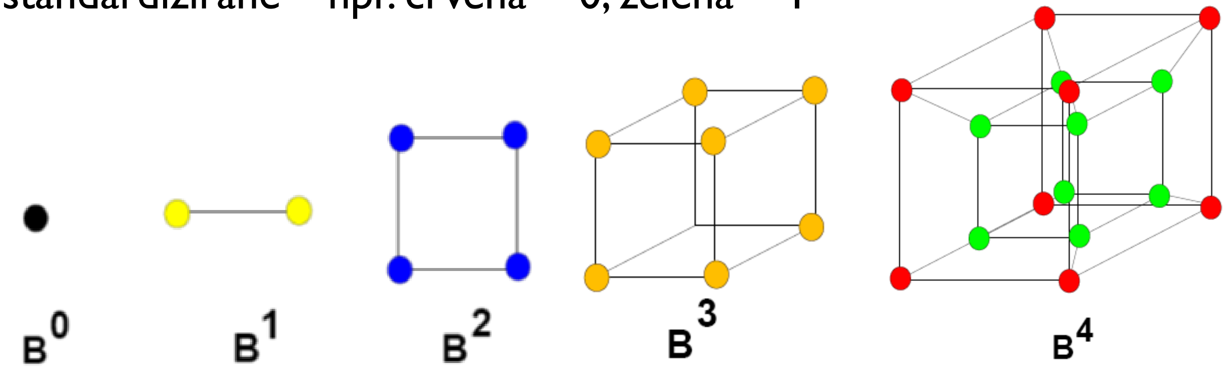
3. **Suma produkata (SOP) ili produkt suma (POS)**
    - SOP – npr. f = $x_1 x_2' + x_1' x_3$
    - POS – npr. f = $(x + y) (z + w)$
    - Konverzija kanonskih oblika (sume mintermi ili produkta makstermi) u SOP ili POS radi se primjenom aksioma i teorema Booleove algebre
    - Prednost: kompaktan opis
    - Nedostatak: nije kanonski, nije nužno minimalan, konverzija SOP ⇔ POS vrlo skupa, skupo određivanje ekvivalencije i zadovoljivosti

4. **Nestandardni (višerazinski opis)**
    - Npr. f = $(xy + uv) (x'y'+ u'w')$
    - Prednost: vrlo kompaktan opis
    - Nedostatak: nije kanonski, računalno vrlo skupa pretvorba u kanonske oblike, skupo određivanje ekvivalencije i zadovoljivosti

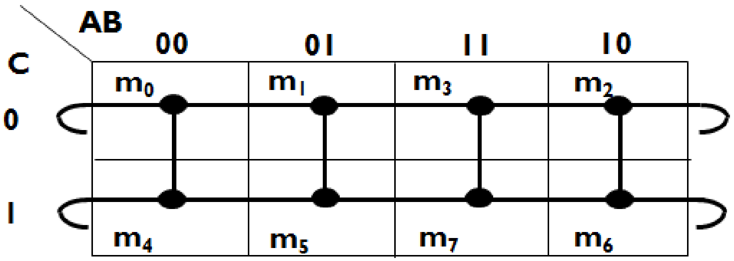
Booleove kocke i *K*-tablice

5. **Booleova kocka** je geometrijsko tijelo koje predstavlja Booleovu funkciju. Svaki vrh je jedan minterm funkcije, a vrh je obojen sukladno vrijednosti funkcije (0 ili 1) za taj minterm

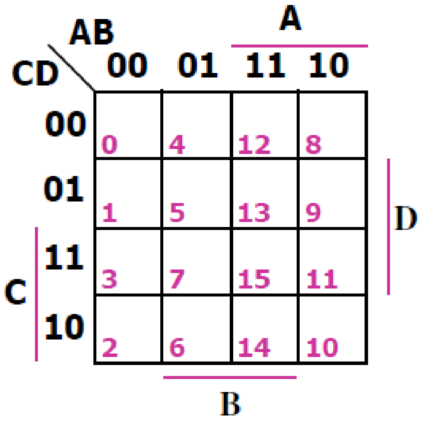
   Boje nisu standardizirane – npr. crvena = 0, zelena = 1



Dvodimenzionalni prikaz Booleove kocke = **Karnaughova (K) tablica**:



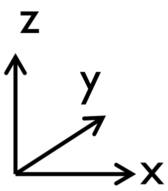
C \ AB	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6



CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

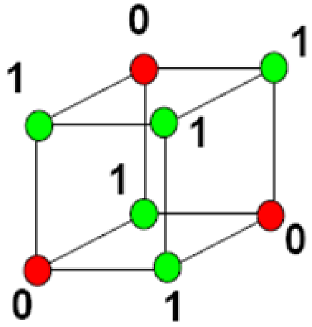
Booleove kocke

- Svaka kombinacija vrhova Booleove kocke je zasebna logička funkcija:

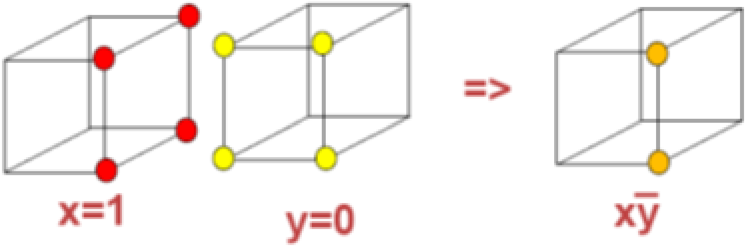


npr.

f = xy’z’ + x’y’z + x’yz’ + xy’z + xyz odgovara kocki:



- Operacije nad literalima se također mogu smatrati operacijama nad Booleovim kockama:
- Npr. *C* = *x y’*

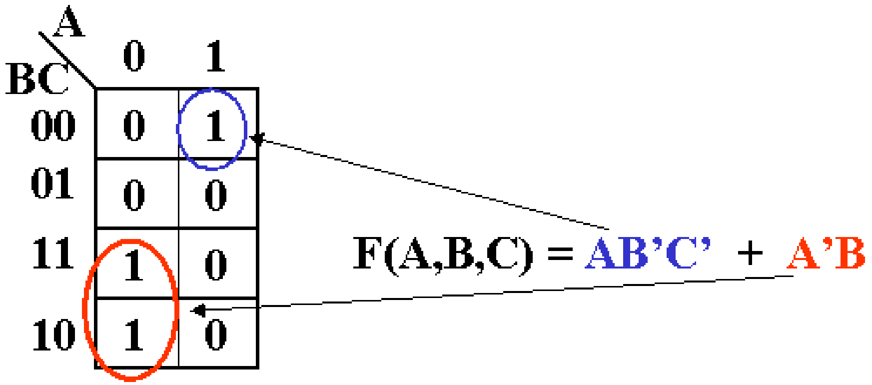


Booleove kocke

- Predstavljanje funkcije kao SOP može se zamisliti kao skup Booleovih kocki

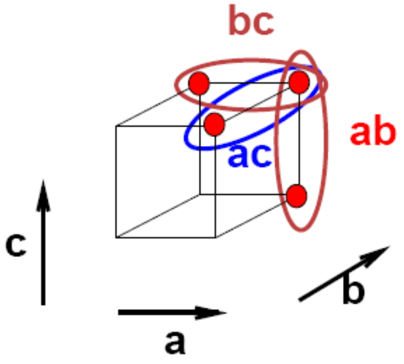
  npr. $f = ab + ac + bc$, $f = \{ab, ac, bc\} = C$

- Skup Booleovih kocki koje predstavljaju logičku funkciju naziva se **pokrivanje** (engl. *cover*, *C*) funkcije *f*
- Svaki vrh s vrijednosti funkcije 1 pokriven je barem jednom kockom
- Postupak dvorazinske minimizacije traži najmanji broj kocki pokrivanja (slično kao i minimizacija *K*-tablicom u 2D prikazu)



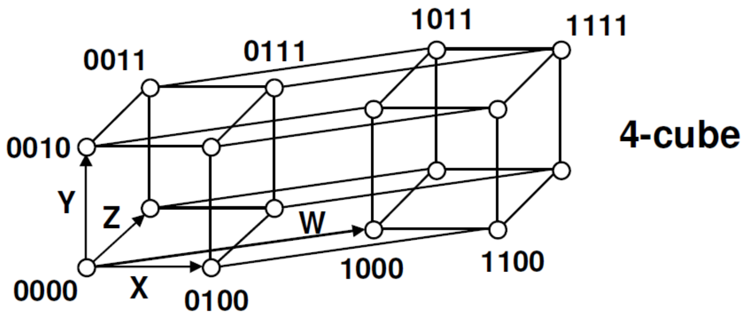
BC \ A	0	1
00	0	1
01	0	0
11	1	0
10	1	0

$F(A,B,C) = AB'C' + A'B$



Booleove kocke

- Prednosti: kanonski, jednostavna provjera ekvivalencije, vizualno razumljivo predstavljanje logičkih funkcija
- Nedostatak: za n > 4 predstavljanje postaje neprimjereno



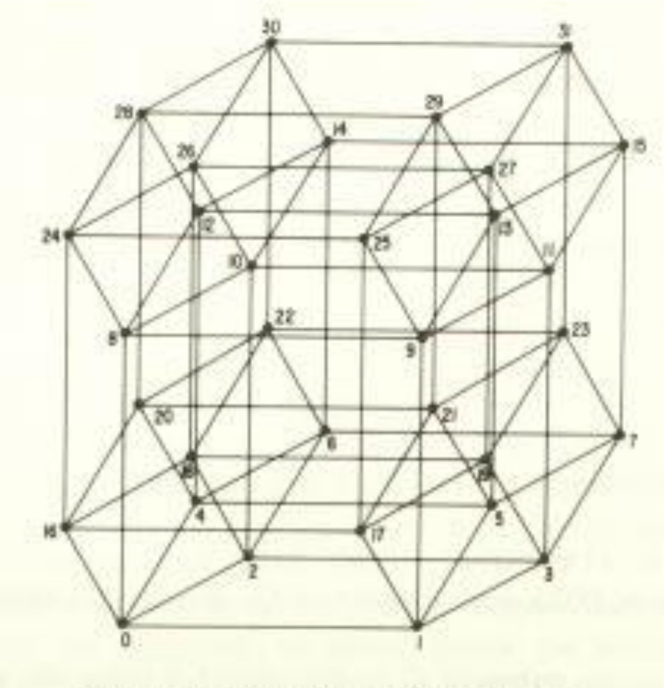


Fig. 1.15 Frisiani's rendition of 5-cube

Primjena predstavljanja Booleovih funkcija: analiza dosezljivosti

- U analizi dosezljivosti zanima nas **skup stanja dostupan iz početnih stanja Q_0 u jednom ili više koraka**. Na početku definiramo funkcije koje će nam dati skup stanja dosezljiv u **jednom koraku** (sljedeća stanja, engl. *next state*).
- Neka je dan **FSM = $(Q, \Sigma, \Delta, \delta, Q_0, \gamma)$**, gdje je Q – skup stanja, Σ – skup ulaznih vrijednosti, Δ – skup izlaznih vrijednosti, δ – funkcija sljedećeg stanja (δ za neki ulaz $x \in \Sigma$ i stanje $s \in Q$ daje jedno sljedeće stanje $t \in Q$), Q_0 – skup početnih stanja, γ – funkcija izlaza.

- Definiramo pomoćnu **logičku** funkciju malo "eta" $\eta(s, t)$:

  $\eta(s, t) = 1$ ***akko*** $\exists_{x=\Sigma} \mid \delta(s, x) = t$

  $\eta(s, t) = 0$ ***inače*** (postoji li ili ne $(s, x, t) \in \delta$)

- Zatim definiramo funkciju H (veliki "eta") koja će dati **skup stanja T $\{t \in Q\}$ dostupnih u jednom koraku** iz skupa zadanih stanja $S \subseteq Q$:
- $H : 2^Q \rightarrow 2^Q$, **H** je funkcija nad svim podskupovima od **Q**, također $H(\emptyset) = \emptyset$.

  $H(S) = \{t \in Q \mid \exists_{s \in S} \eta(s, t)\}$ ***= sva sljedeća stanja***

Kodiranje FSM-a

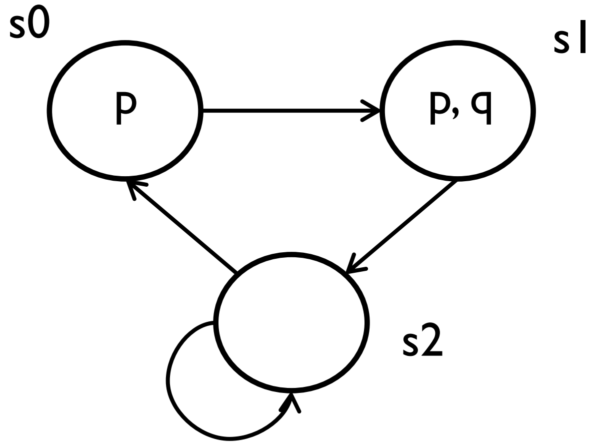
- Uobičajene oznake za $FSM = (Q, \Sigma, \Delta, \delta, Q_0, \gamma)$ preslikavaju se u oznake koje označuju **kodirane skupove**.
- $FSM_{KODIRANO} = (S, I, O, N, S_0, Y)$

- Uz $B = \{0, 1\}$, kodiranje daje:

$S = B^n$	skup stanja (***n*** = broj bitova za kodiranje)
$I = B^m$	skup ulaznih vektora od ***m*** bitova
$O = B^p$	skup izlaznih vektora od ***p*** bitova
$N : B^n \times B^m \rightarrow B^n$	funkcija sljedećeg stanja
$S_0 \subseteq S$	skup početnih stanja
$Y : B^n \times B^m \rightarrow B^p$	funkcija izlaza

- **Time se sve značajke FSM-a mogu izraziti preko Booleovih logičkih funkcija**

Primjer kodiranja Kripkeove strukture M(S,R,L)



S = {s0, s1, s2}

R = {(s0,s1),(s1,s2), (s2,s2), (s2,s0)}

L(s0) = {p}

L(s1) = {p,q}

L(s2) = {}

S_{enc} = {(1,0), (1,1), (0,0)}

R_{enc} = {(1,0,1,1), (1,1,0,0),(0,0,0,0), (0,0,1,0)}

f_{Senc}=pq’ + pq + p’q’

f_{Renc}=p1q1’p2q2 + p1q1p2’q2’ + p1’q1’p2’q2’ + p1’q1’p2q2’

Rješenje

- Potrebno je uvesti novu matematičku strukturu koja će omogućiti učinkoviti zapis karakterističnih Boolevih funkcija u računalu
- Takvom strukturom omogućit će se učinkovitija analiza dosezljivosti pomoću razmatranja skupova stanja, a ne samo pojedinačnih stanja
- Za izgradnju takve strukture u računalu koristit će se posebno oblikovani algoritam

Zadatci

1. Za varijablu programa zadanu kao podskup cijelih brojeva
   S = {3..11} odredite njezinu karakterističnu Booleovu funkciju

2. Naći karakterističnu funkciju relacije
   $R = \{(s, t) \in (S \times T) \mid t = s + 1\}$, pri čemu je:
   $S = \{2, 4, 6, 8\}$, $T = \{1, 5, 7, 9, 10\}$

BINARNI DIJAGRAMI ODLUČIVANJA

Što tražimo od prikaza Booleovih funkcija?

- Kanonski (jedinstveni način prikaza)
- Što manje zauzeće vremenskih i prostornih resursa u računalu
- Efikasne operacije provjere ekvivalentnosti, tautologije, zadovoljivosti formule i sl.
- Efikasne operacije verifikacije provjerom modela (dostupnost stanja, izračun "čvrste točke", i sl.)
- Sve navedene ciljeve zadovoljava predstavljanje logičkih funkcija **binarnim dijagramima odlučivanja (BDD)**

Kofaktori i teorem ekspanzije

- **Shannonovi (Booleovi) kofaktori**
  - Neka je $f(x) : B^n \rightarrow B$, $x = (x_1, \dots x_n)$ logička, Booleova funkcija
  - **Kofaktori** f_x (pozitivni) odnosno $f_{x'}$ (negativni) funkcije f po varijabli x_i :

$$f_{x_i}(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$
$$f_{x_i'}(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

  - To je funkcija f u kojoj je fiksirano $x_i = 1$ $(x_i' = 0)$, odnosno $x_i = 0$ $(x_i' = 1)$.

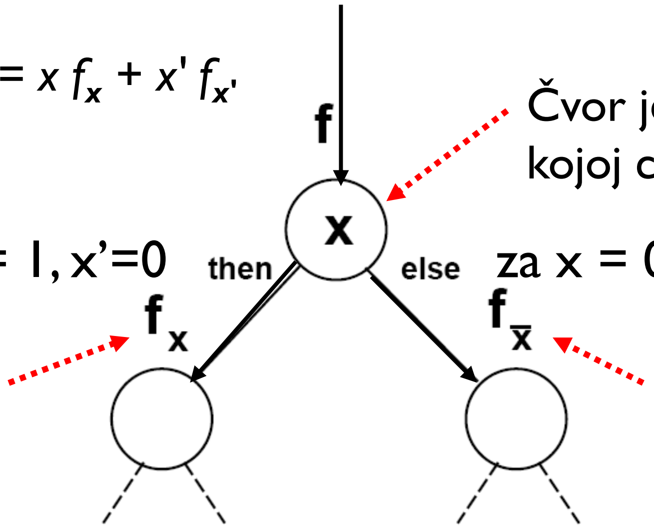
- **Shannonov teorem ekspanzije**
  - Svaka logička funkcija može se ekspandirati oko varijable x_i u SOP oblik:

$$f(x) = x f_x + x' f_{x'}$$

  - Varijabla x_i naziva se **varijabla cijepanja** (engl. *splitting variable*).
  - Pri tome su f_x i $f_{x'}$ kofaktori

Prikaz Shannonove ekspanzije grafom

- Cijepanje po varijabli *x* prikazuje se **grafom** (binarnim stablom):



$\boldsymbol{f} = x\, f_x + x'\, f_{x'}$

Čvor je varijabla po kojoj cijepamo

za x = 1, x'=0

za x = 0, x'=1

kofaktor po x

kofaktor po x'

Najčešća uporaba **oznaka na lukovima**: ***1, 0*** ili ***T, E*** (od Then, Else)

Binarni dijagram odlučivanja (BDD – engl. *Binary Decision Diagram*)

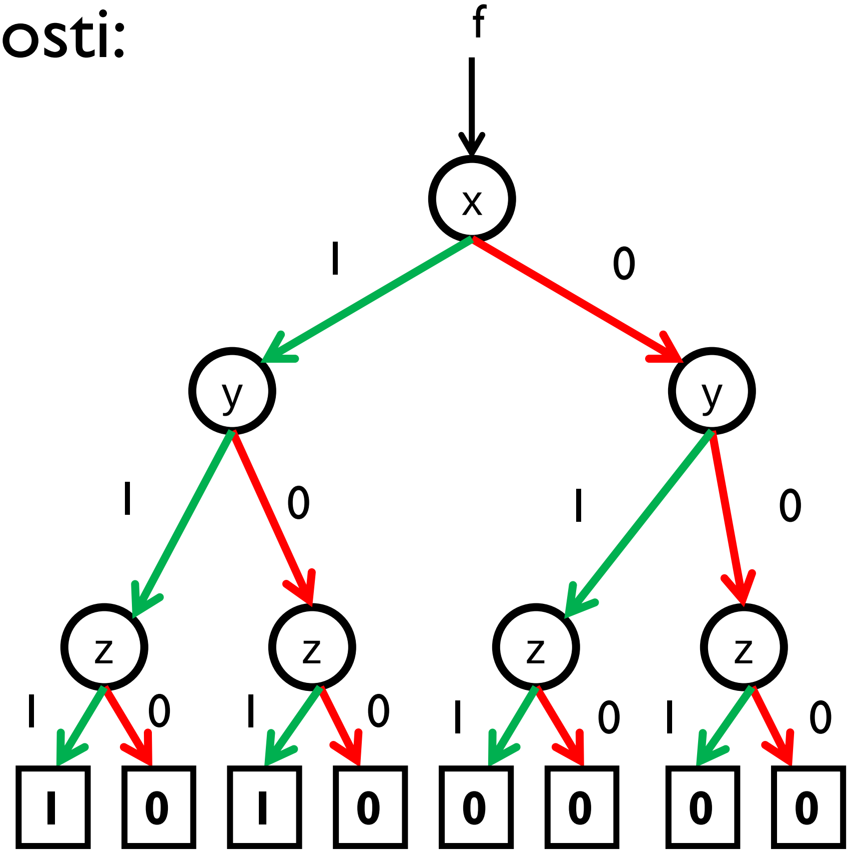
- **G(V, E)** – Usmjereni, aciklički graf (DAG, engl. *Directed Acyclic Graph*)
  - ***V*** – skup čvorova, **nezavršni** čvorovi označuju **varijable** $x_i \in X$, **završni** čvorovi označeni su **konstantama** 0 ili 1.
  - ***E*** – skup **usmjerenih** lukova $E \subseteq V \times V$, označenih **1 (then, T)** ili **0 (else, E)**.
  - *X* – **uređen skup varijabli**. *rank(v)* označuje **redoslijed** oznake čvora *v*.
  - **Svaki put u grafu slijedi zadanu uređenost varijabli.**
  - Svaki **nezavršni čvor** $\mathbf{x_i}$ predstavlja neku logičku funkciju koja se cijepa po toj varijabli:

$f = x\, f_x + x'\, f_{x'}$, gdje su kofaktori: $f_x = f\,|_{x=1}$, $f_{x'} = f\,|_{x=0}$ izravni sljedbenici čvora $\mathbf{x_i}$ na lukovima **then** (**T**, **1**) i **else**(**E**, **0**).

BDD

- Primjer BDD-a za funkciju *f* zadanu tablicom istinitosti:

x	**y**	**z**	**f**
0	**0**	**0**	**0**
0	**0**	**1**	**0**
0	**1**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**1**	**1**



ROBDD – smanjeni, uređeni binarni dijagrami odlučivanja

Engl. Reduced Ordered Binary Desision Diagram

- Smanjenje i uređenje BDD-a očituje se u tome da:
  - Ne postoji čvor *v* takav da then(*v*) = else(*v*), tj. čvor nema jednaku djecu
  - Izomorfni podgrafovi (podgrafovi s istom strukturom i označavanjem) se svode na jedan podgraf.
  - Crtaju se samo dva završna čvora, 1 i 0, a ne dva završna za svaki nezavršni čvor na zadnjoj razini grafa kao kod „običnog” BDD-a

Postupak izgradnje ROBDD-a iz funkcije zadane tablicom istinitosti

1. Nacrtaj čitavi BDD, označi istinitosti u završnim čvorovima
2. Sad nacrtaj samo dva završna čvora (1 i 0), preusmjeri sve veze od zadnjih nezavršnih čvorova u odgovarajuće završne čvorove
3. Kretajući se od razine zadnjih nezavršnih čvorova prema korijenu grafa, najprije ukloni sve čvorove koji imaju **then**(v) = **else**(v), a u drugom prolazu sve izomorfne podgrafove zamijeni s jednim na istoj razini

Primjer 1: Nacrtaj ROBDD za funkciju f

uz uređenje: x < y < z:

Najprije varijabla x, pa y, pa z

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

ROBDD - kanoničnost

- ROBDD predstavljaju **kanonski (jedinstveni) oblik** logičke funkcije (Bryant, 1986.)

Posljedice kanoničnosti:

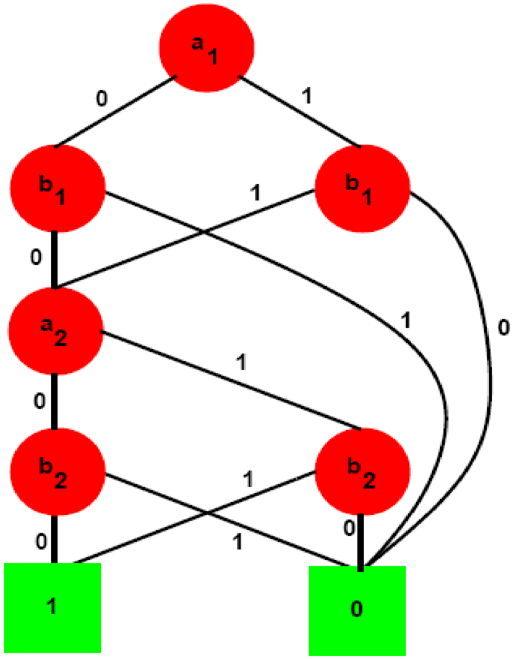
- Dvije logičke funkcije su **ekvivalentne** ako su njihovi ROBDD grafovi izomorfni. Izomorfnost je postignuta ako postoji bijekcija između grafova takva da se završni čvorovi preslikavaju u završne, a nezavršni u nezavršne s istim vrijednostima čvorova djece. Uvjet je **jednaka uređenost** varijabli.
- Završni čvor **1** predstavlja logičku funkciju **f = 1**.
- Završni čvor **0** predstavlja logičku funkciju **f = 0**.
- Odlučivanje o ekvivalentnosti logičkih formula i zadovoljivosti, koja se svodi na pokazivanje ne-ekvivalencije s grafom funkcije f = 0, su bitno pojednostavljeni.

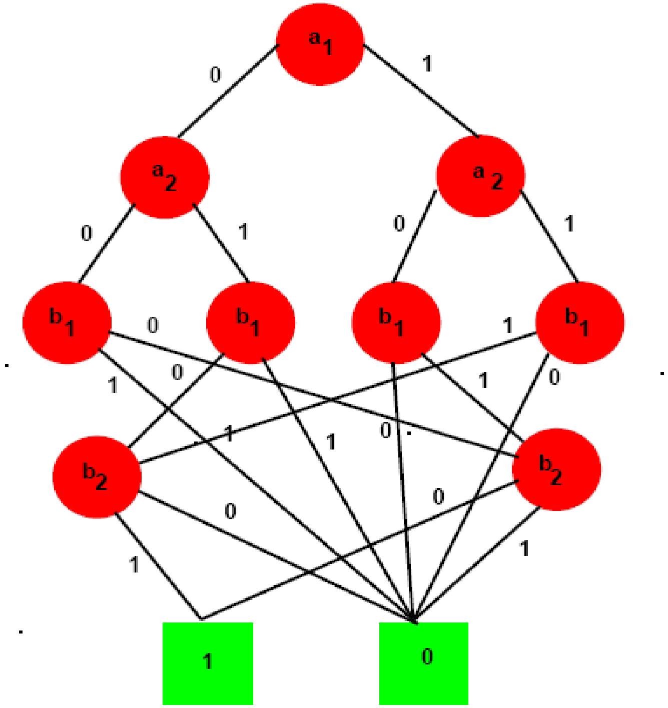
ROBDD – uređenost varijabli

Složenost ROBDD-a **značajno ovisi o uređenosti** (redoslijedu) varijabli.

Neke funkcije imaju eksponencijalno velike ROBDD-ove za svaku uređenost varijabli, dok neke imaju polinomijalno (pa i linearno) velike BDD-ove za neka uređenja, a eksponencijalno za druga. **Određivanje optimalne uređenosti – NP-težak problem.**

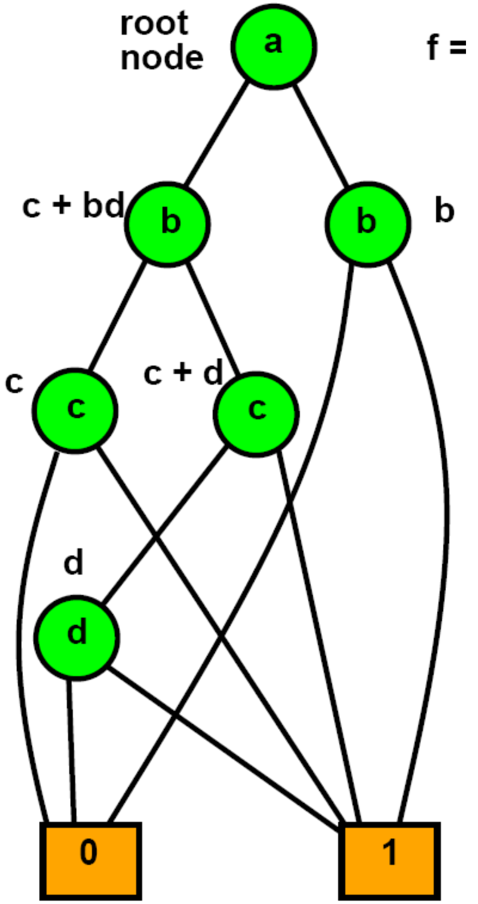
Primjer dvobitnog komparatora: Na lijevoj slici uređenost ($a_1 < b_1 < a_2 < b_2$), na desnoj slici uređenost ($a_1 < a_2 < b_1 < b_2$)





Učinkovitost prikaza funkcija BDD-ovima

- ROBDD ne numerira eksplicitno putove (kao tablica), već graf s **linearnim brojem čvorova** predstavlja **eksponencijalan broj putova**
- ROBDD-ovi se u praksi koriste kad želimo za neki problem pronaći **skup svih mogućih rješenja**
- Za razliku od ROBDD-a, SAT-rješavači (tema idućeg predavanja) se koriste samo za vrlo učinkovit pronalazak jednog rješenja (ako ono postoji)



ALGORITAM ITE I IMPLEMENTACIJA BDD-OVA

Operator ITE

- U izgradnji ROBDD-a ponavlja se postupak: odabere se varijabla iz uređenog skupa i funkcija se cijepa sukladno Shannonovom teoremu.
- Za implementaciju izgradnje ROBDD-a u računalu, potrebno je pronaći pogodnu **funkciju (operator)** za navedeni postupak i izgraditi algoritam njezine uporabe.
- Jedna takva pogodna funkcija s **tri argumenta** je Booleova funkcija (operator) ***ite***:

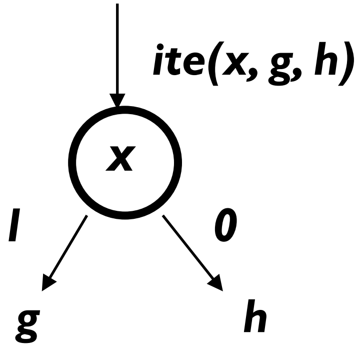
$$ite(f, g, h) = f\, g + f'\, h$$

- gdje su ***f, g, h*** proizvoljne logičke (Booleove) funkcije. Ta se funkcija interpretira ***(i-t-e)***:
- **I**F *f* (tj. ako *f=1*) **T**HEN *g*, **E**LSE (tj. ako *f=0*) *h*.
- *To je generalizacija sklopa multipleksor (u sklopu multipleksor uobičajeno funkcije f, g, h su pojedinačne Booleove varijable).*

Odnos ITE-a i BDD-ova

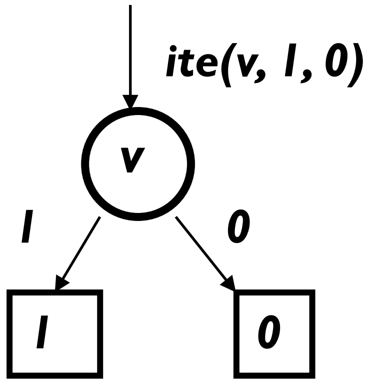
- Neka je ***x*** Booleova varijabla.
- $f = x$ je jedna logička funkcija.
- Ako to uvrstimo u ***ite*** operator slijedi:

  $\mathit{ite}(x, g, h) = x g + x' h$
- Izraz predstavlja cijepanje po varijabli ***x*** (sukladno Shannonu) što možemo prikazati grafički:



- Slijedi: ako ***ite*** operator ima kao **prvi parametar samostalnu (vršnu) varijablu**, tada ***ite*** predstavlja BDD s čvorom te varijable, te lijevim lukom (1, T) koji završava u funkciji **g**, a desnim (0, E) u funkciji ***h***.

- Neka je ***v*** Booleova varijabla.
- $f = v$ (**funkcija jedne varijable**) može se predstaviti operatorom ***ite*** kao: ***ite(v, 1, 0)***
- Odnosno grafički:

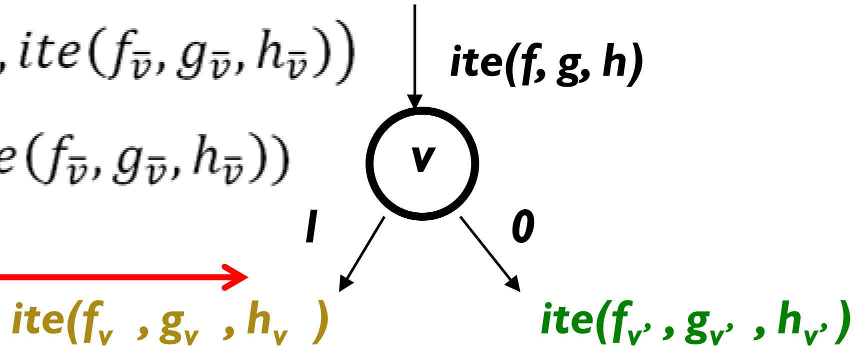


Rekurzija u ITE

- Neka su općenite logičke funkcije *f*, *g*, *h* zadane svaka sa svojim ROBDD-om.
- Neka je ***ite(f, g, h)* = *(f g + f' h)*** funkcija triju argumenata (logičkih funkcija) te neka je varijabla ***v*** prva u nizu uređenih varijabli svih triju funkcija (*f*, *g*, *h*). Rastavljanje *ite(f, g, h)* po Shannonu pokazuje **rekurziju**:

$$
\begin{aligned}
ite(f,g,h) &= fg + \bar{f}h \\
&= v\left(fg + \bar{f}h\right)_v + \bar{v}\left(fg + \bar{f}h\right)_{\bar{v}} \\
&= v\left(f_v g_v + \bar{f}_v h_v\right) + \bar{v}\left(f_{\bar{v}} g_{\bar{v}} + \bar{f}_{\bar{v}} \bar{h}_{\bar{v}}\right) \\
&= ite\left(v, ite(f_v, g_v, h_v), ite(f_{\bar{v}}, g_{\bar{v}}, h_{\bar{v}})\right) \\
&= \left(v, ite(f_v, g_v, h_v), ite(f_{\bar{v}}, g_{\bar{v}}, h_{\bar{v}})\right)
\end{aligned}
$$

To je **ROBDD** s vršnom varijablom ***v***



Ostvarivanje Booleovih operacija ITE-om

- **ITE-operator jednostavno implementira svih 16 logičkih funkcija s dvije varijable:**

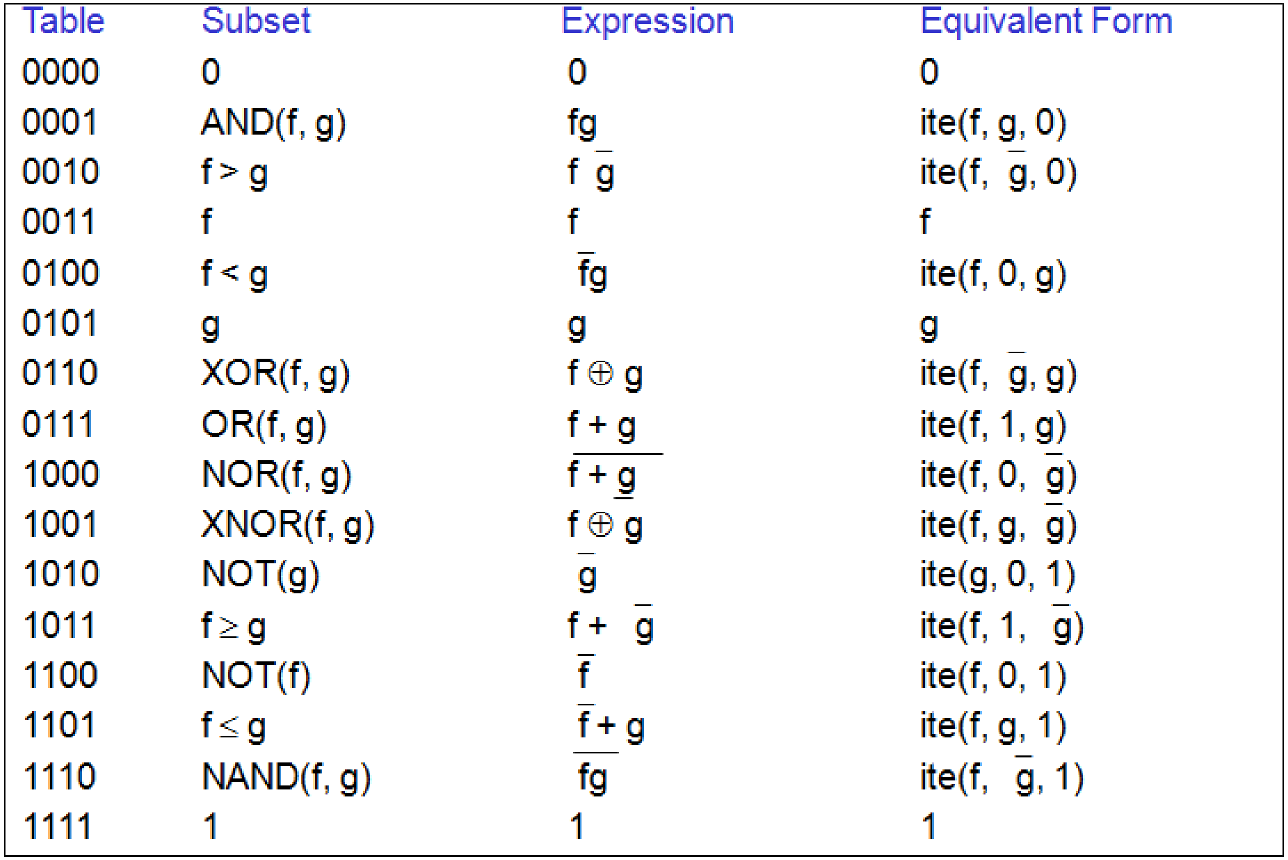


Table	Subset	Expression	Equivalent Form
0000	0	0	0
0001	AND(f, g)	fg	ite(f, g, 0)
0010	f > g	$f\bar{g}$	ite(f, $\bar{g}$, 0)
0011	f	f	f
0100	f < g	$\bar{f}g$	ite(f, 0, g)
0101	g	g	g
0110	XOR(f, g)	$f \oplus g$	ite(f, $\bar{g}$, g)
0111	OR(f, g)	f + g	ite(f, 1, g)
1000	NOR(f, g)	$\overline{f + g}$	ite(f, 0, $\bar{g}$)
1001	XNOR(f, g)	$f \oplus g$	ite(f, g, $\bar{g}$)
1010	NOT(g)	$\bar{g}$	ite(g, 0, 1)
1011	f ≥ g	$f + \bar{g}$	ite(f, 1, $\bar{g}$)
1100	NOT(f)	$\bar{f}$	ite(f, 0, 1)
1101	f ≤ g	$\bar{f} + g$	ite(f, g, 1)
1110	NAND(f, g)	$\overline{fg}$	ite(f, $\bar{g}$, 1)
1111	1	1	1

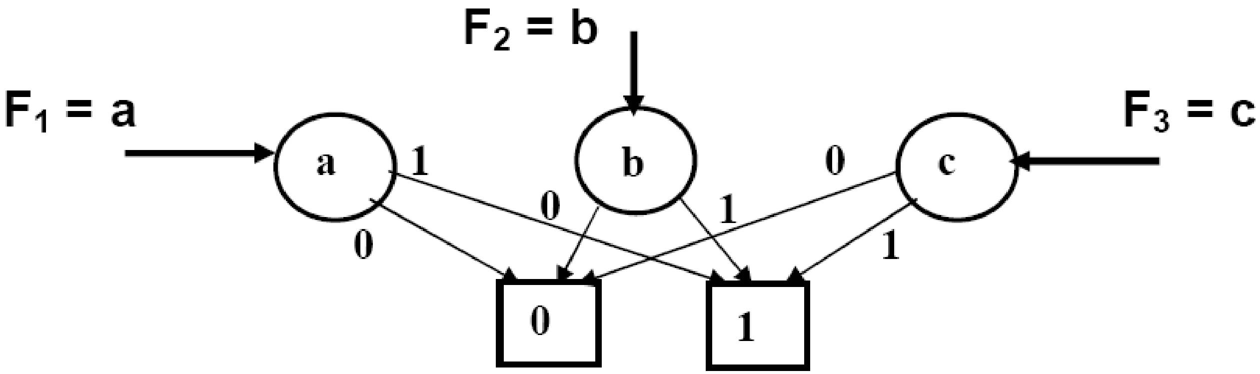
Primjer izgradnje ROBDD-a ITE-algoritmom

- Izgradi ROBDD za funkciju: ***F = ac + bc***

  uz uređenost varijabli: *a < b < c (prva a, zadnja c)*.

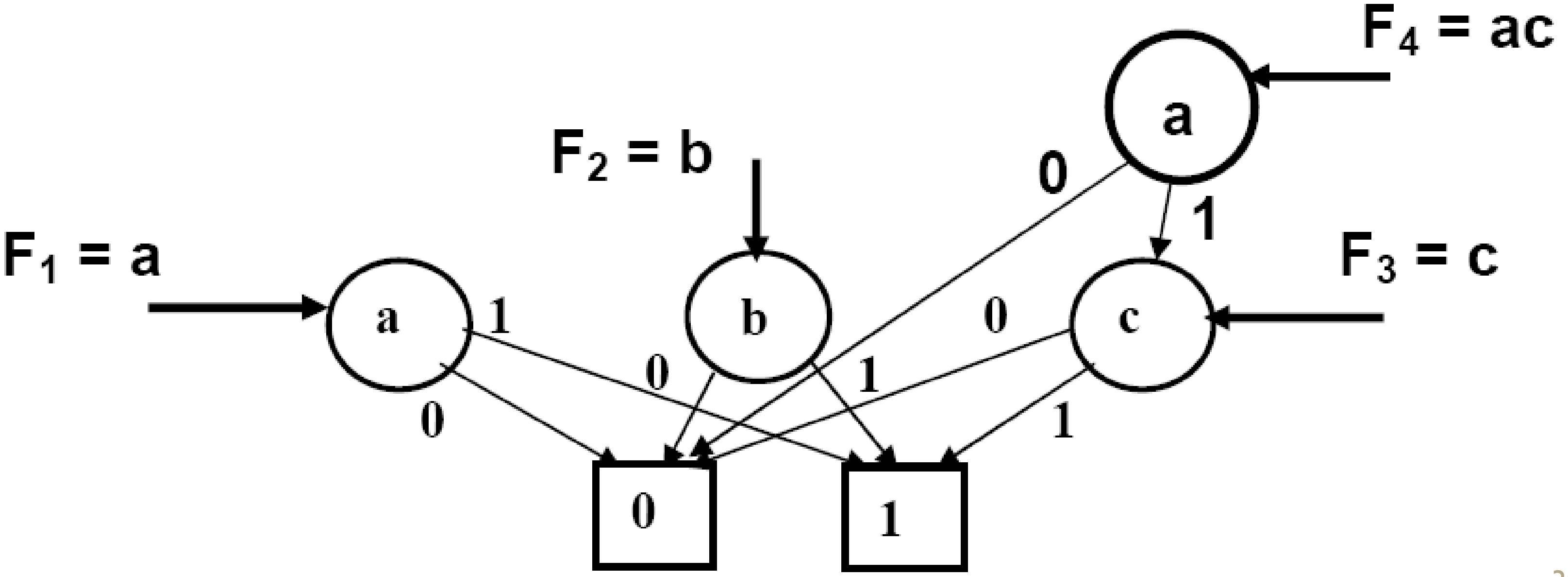
- **<u>1. korak</u>**: izgradnja elementarnih ROBDD-ova za svaku varijablu uporabom ***ite(var, 1, 0)***.

- Čvorovi ***a***, ***b***, ***c*** predstavljaju korijene ROBDD-ova projekcijskih funkcija (funkcija jedne varijable) $\mathbf{F_1 = a}$, $\mathbf{F_2 = b}$, $\mathbf{F_3 = c}$. Završni elementarni čvorovi **0** i **1** su zajednički (izomorfni grafovi).



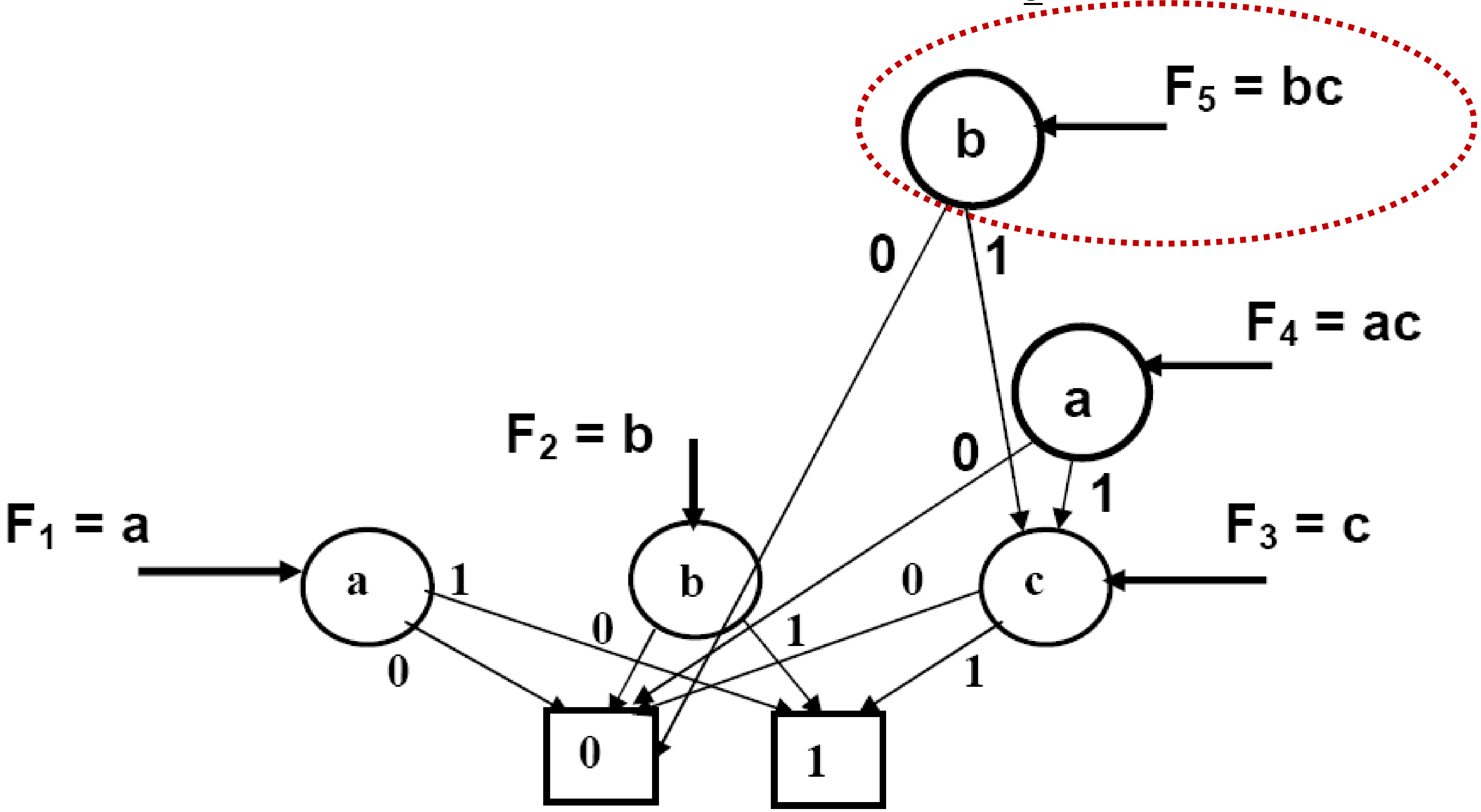
Primjer izgradnje ROBDD-a ITE-algoritmom

- **2. korak**: za produkt ***(ac)*** uporabom ***(ac)* = *a* $\wedge$ *c* =** ***ite(a, c, 0)*** . Varijabla ***a*** je **samostalna**. To je dijagram s **vršnom varijablom *a***, THEN stranom ***c***, ELSE stranom ***0***. Čvorovi za THEN stranu ***c*** i ELSE stranu ***0*** već postoje. U dijagram se dodaje **novi čvor *a*** koji se povezuje na već postojeći korijenski čvor funkcije ***F_3=c*** i završni čvor ***0***.
- Taj novi čvor je korijen ROBDD-a za funkciju ***F_4 = ac***.



Primjer izgradnje ROBDD-a ITE-algoritmom

- **3. korak**: za produkt ***(bc)*** uporabom ***(bc) = b*** $\wedge$ ***c = ite(b, c, 0)*** . To je dijagram s **vršnom varijablom *b***, THEN stranom ***c***, ELSE stranom ***0***. Čvorovi za THEN stranu ***c*** i ELSE stranu ***0*** već postoje. U dijagram se dodaje **novi čvor *b*** koji se povezuje na postojeći korijenski čvor F_3**=c** i ***0***. Taj novi čvor je korijen ROBDD-a za funkciju ***F_5 = bc***.



Primjer izgradnje ROBDD-a ITE-algoritmom

- **4. korak**: izgradnja ROBDD-a za cijelu funkciju $F = ac + bc$, uporabom $(ac + bc) = (ac \vee bc) = ite(ac, 1, bc)$.
- Vršna varijabla u ***ite*** funkciji **nije samostalna** pa je potrebno rastavljanje ***ite*** funkcije rekurzijom pazeći na **uređenost** varijabli.

  $$ite(f, g, h) = a\, ite(f_a , g_a , h_a) + a'\, ite(f_{a'} , g_{a'} , h_{a'}) =$$
  $$= ite[a, ite(f_a , g_a , h_a), ite(f_{a'} , g_{a'} , h_{a'})]$$

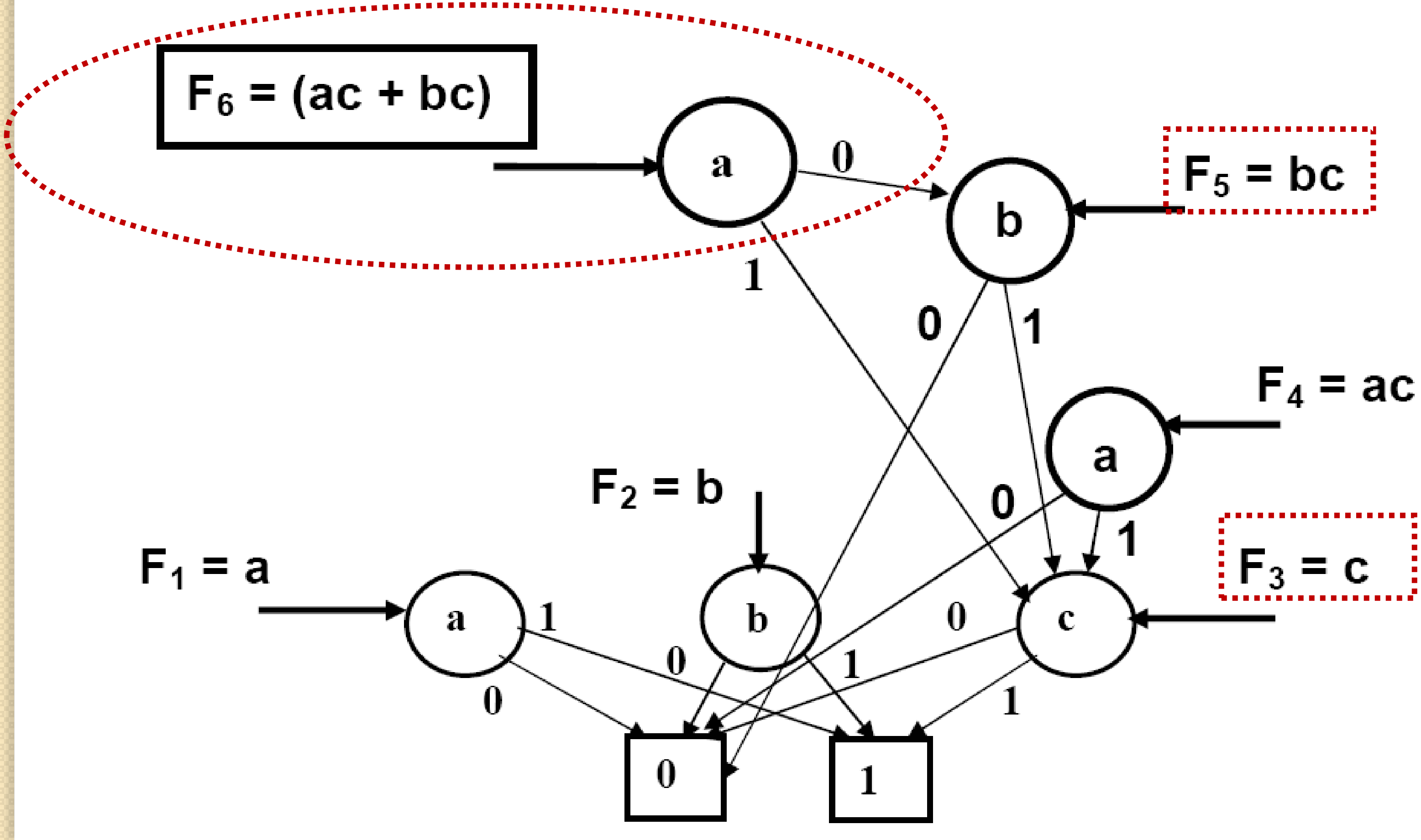
- U našem primjeru: $f=ac, g=1, h=bc$.
- Kofaktori tih funkcija za varijablu ***a*** su: $f_a = c,\quad g_a = 1,\quad h_a = bc$
  $f_{a'} = 0,\quad g_{a'} = 1,\quad h_{a'} = bc$
- Što rezultira u: $ite(ac, 1, bc) = ite[a, ite(c, 1, bc), ite(0, 1, bc)]$
- Neke ROBDD-ove za ***ite*** funkcije možemo pojednostaviti:

  $ite(c, 1, bc) = c$, jer to znači ako $c=1$ THEN 1, ELSE $c=0$ pa je $bc=0$

  $ite(0, 1, bc) = bc$, jer to znači ako 0, onda to nužno mora biti bc.
- Konačno: $ite(ac, 1, bc) = ite(a, c, bc)$
- To je ROBDD s korijenskim čvorom ***a***, T lukom u funkciji ***c*** i E lukom u ***bc***.

Primjer izgradnje ROBDD-a ITE-algoritmom

- **5. korak**: izgradnja ROBDD-a za cijelu funkciju ***F = ac + bc***, uporabom ***ite(a, c, bc)*** sastoji su u dodavanju **novoga čvora *a*** koji se povezuje na postojeće korijenske čvorove funkcija ***F_3 = c*** i ***F_5 = bc***. Taj čvor ***a*** je korijenski čvor ROBDD-a za zadanu funkciju ***F_6 = ac + bc***.



Zaključci o postupku izgradnje ROBDD-a

- Primjer izgradnje ROBDD-a pokazao je da:

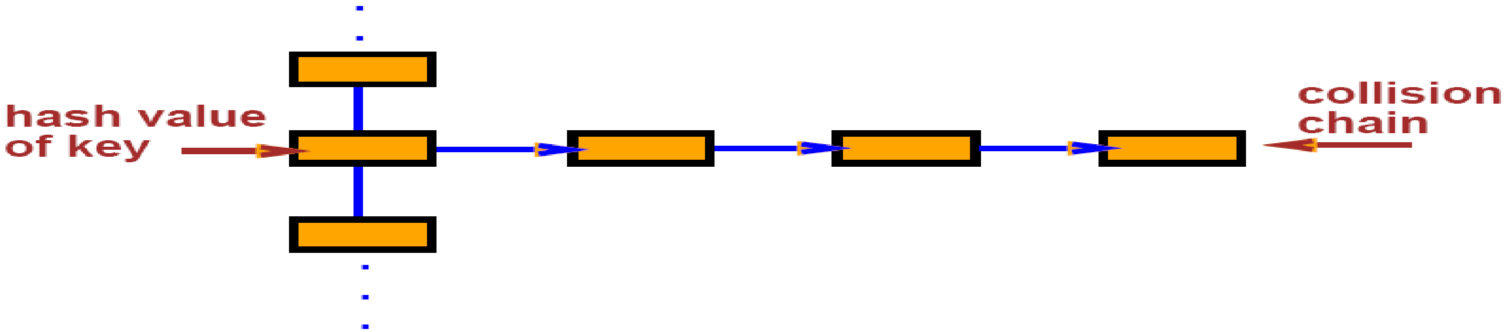
1. Svaki čvor u ROBDD-u predstavlja korijenski čvor ROBDD-a jedne jedinstvene logičke funkcije.
2. Najprije se grade jednostavni ROBDD-i za krajnje funkcije (npr. za varijable a, b, c u primjeru)
3. Tijekom izgradnje ROBDD-a za neku novu funkciju, potrebno je u svakom koraku provjeriti postoji li već možda traženi ROBDD za jednostavniju funkciju (kofaktor) s odgovarajućim korijenskim čvorom. Da bi se to omogućilo, **svaki čvor s podstablom u ROBDD-u mora biti jedinstveno označen**.

ROBDD – jedinstvena tablica

- Algoritmi izgradnje ROBDD-ova pohranjuju **sve čvorove** (***v*, *g*, *h***), gdje je ***v*** oznaka (indeks) vršne varijable u čvoru, a ***g*** i ***h*** su jedinstveni opisi (indeksi ili pokazivači) THEN i ELSE lukova iz te varijable. Potpuna struktura opisa čvora pohranjuje se u **jedinstvenoj tablici** (engl. *unique table*).
  - Ako traženi čvor već postoji u tablici, u daljnjoj izgradnji koristi se indeks ili pokazivač (engl. *pointer*) na taj čvor u jedinstvenoj tablici.
  - Ako traženi čvor ne postoji u tablici, zapisuje se taj čvor u jedinstvenu tablicu i vraća se novi indeks (pokazivač).
  - Time se ostvaruje kanonski zapis, jer za neki čvor ***f* = (*v*, *g*, *h*)** postoji **jedan i samo jedan** indeks (pokazivač), a to je adresa strukture opisa toga čvora u jedinstvenoj tablici.
  - Za efikasniju pohranu u memoriji računala na indekse vršnih varijabli i na indekse njihovih odgovarajućih THEN i ELSE strana primjenjuje se **funkcija "hash"**.

ROBDD – jedinstvena tablica

- Poznato je da funkcija “hash” nije savršena, pa se može dogoditi da indeksi (ključevi, engl. *key*) različitih čvorova rezultiraju u istoj adresi elementa jedinstvene tablice. Stoga se na istoj adresi jedinstvene tablice može formirati **kolizijski lanac** (lanac je implementiran povezanom listom).
- Opis svakog čvora u jedinstvenoj tablici mora se proširiti s eventualnim pokazivačem na sljedeći čvor u kolizijskom lancu.

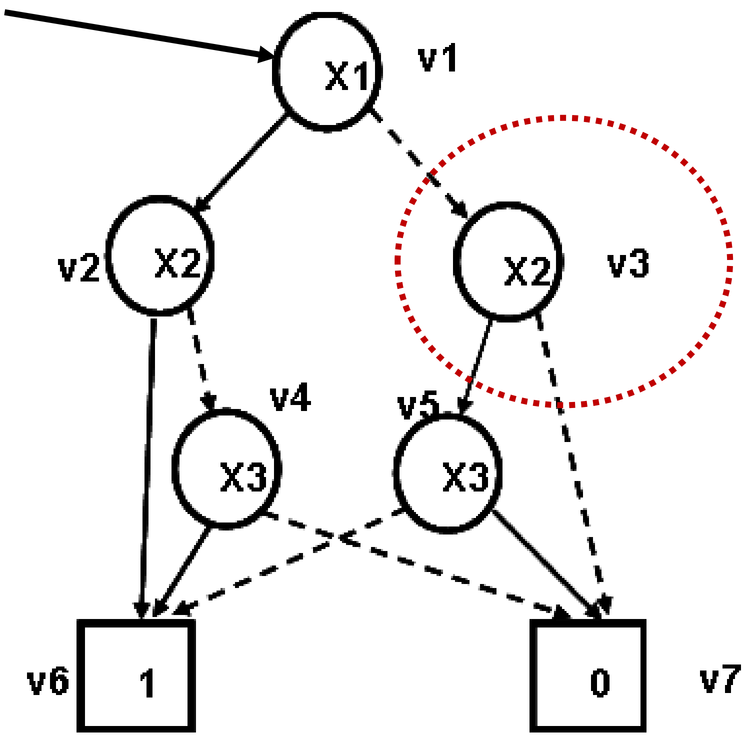


Opis svakog čvora u jedinstvenoj tablici je struktura podataka koja sadrži:

- indeks samostalne varijable
- indeks THEN strane
- indeks ELSE strane
- pokazivač na sljedeću strukturu u kolizijskom lancu

ROBDD – jedinstvena tablica - primjer

- Neka je zadana funkcija ***f* = (x1, x2, x3)** predstavljena ROBDD-om.
- Svaki čvor jedinstveno je određen indeksima varijable $\boldsymbol{x_i}$, THEN strane $\boldsymbol{v_j}$ i ELSE strane $\boldsymbol{v_k}$, te pokazivačem na sljedeću strukturu u kolizijskom lancu.
- Npr. čvor v3 je određen: (x2, v5, v7)



- Jedinstvena tablica je niz od 5 lokacija, (od 0 do 4).
- Moguća "hash" funkcija:

  $\boldsymbol{H(x_i , v_j , v_k) = (i + j + k) \ (mod\ 5)}$

- Npr. za čvor **v3**: $H = 2+5+7\ (mod\ 5) = \mathbf{4}$

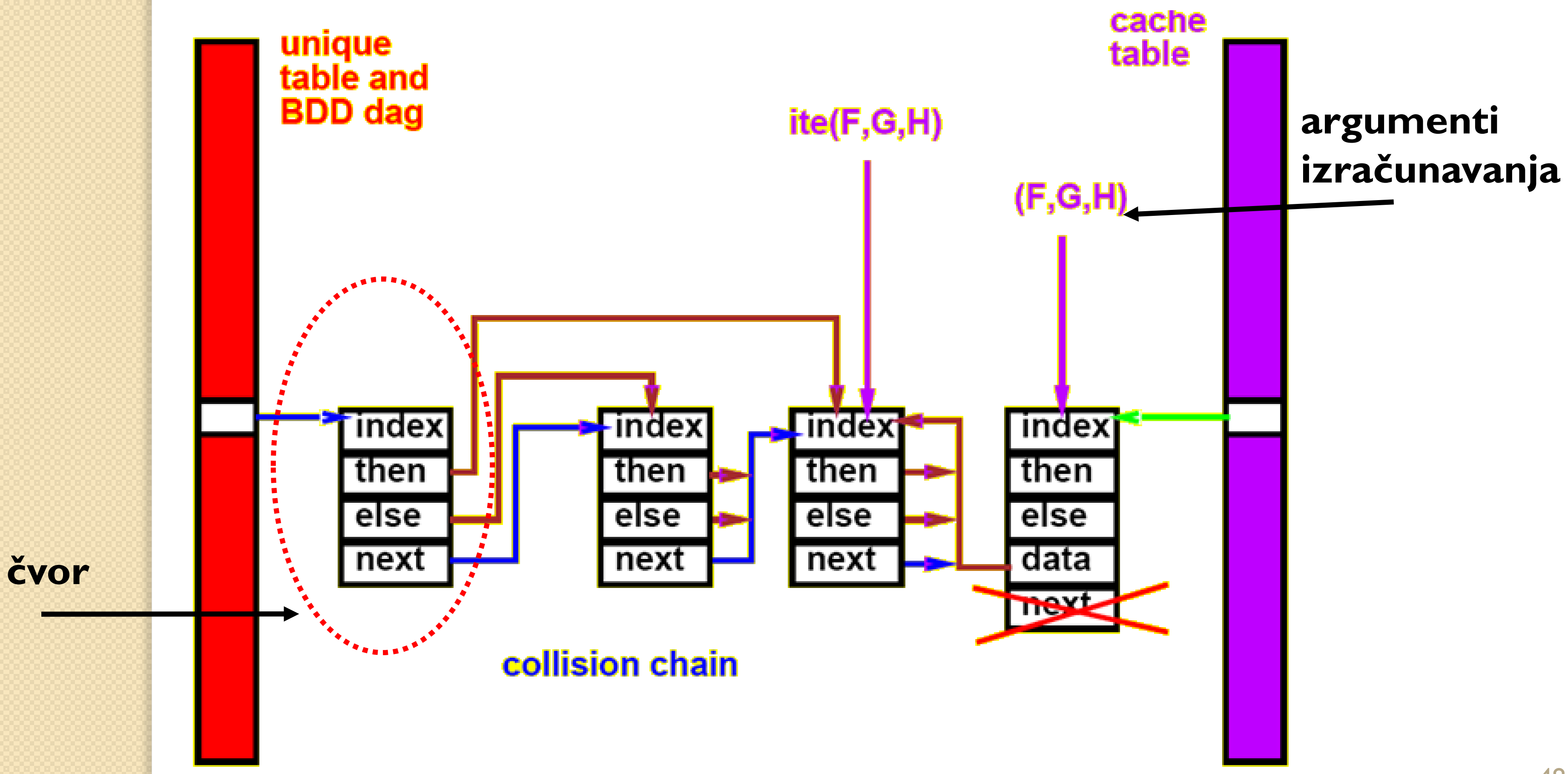
- Čvorovi **v1**, **v4**, **v5** imaju istu vrijednost funkcije "hash" (ostatak=1) pa je za njih potrebno formirati kolizijski lanac.
- Kako bi se smanjio broj kolizija, potrebno je pametno odabrati "hash" funkciju (obično da daje uniformnu razdiobu vrijednosti).

ROBDD – izračunska tablica

- Pored **jedinstvene tablice**, daljnje ubrzanje izgradnje i manipulacije s ROBDD-ovima postiže se uporabom još jedne tablice – **izračunske tablice** (engl. *computed table*), "cache", priručne tablice.
- **Jedinstvena** tablica odgovara na pitanje: "**Postoji li jedinstveni čvor** s vršnom varijablom ***v***, THEN stranom ***g*** i ELSE stranom ***h***, tj. jedinstveni čvor ***(v, g, h)*** ?"
- **Izračunska tablica** odgovara na pitanje: "**Je li nedavno izračunata neka funkcija?**", (npr. konjunkcija dviju logičkih funkcija pomoću ***ite(f, g, h)*** operatora). Time možemo izbjeći ponovno računanje ROBDD strukture koja predstavlja traženi rezultat.
- Rezultat izračunavanja je konačno **jedan ROBDD** pa izračunska tablica privremeno pohranjuje argumente izračunavanja i pokazivač na jednaku strukturu podataka kao i jedinstvena tablica, tj. na čvor ***(v, g, h)***.
- **Bitna razlika: izračunska tablica ne koristi kolizijski lanac – u slučaju kolizije stariji podatak se odbacuje.**

ROBDD – jedinstvena i izračunska tablica

- Budući da pokazuju na istu strukturu, jedinstvena i izračunska tablica mogu spojiti pokazivače na ROBDD čvorove



ITE-algoritam (uz korištenje jedinstvene i izračunske tablice)

Terminal cases:

$$\begin{aligned} (0, g, f) &= (1, f, g) = f \\ ite(f, g, g) &= g \end{aligned}$$

```
$ite(f, g, h)$
    if(terminal case){              /* možda “terminal case”, a ako nije, pogledaj je
        return result;              li to već izračunato u izračunskoj tablici */
    } else if (computed-table has entry $(f, g, h)${
        return result;
    }else{
        let $v$ be the top variable of $(f, g, h)$;
        $\widetilde{f} \leftarrow ite(f_v, g_v, h_v)$;                   /* rekurzivno izračunavanje
        $\widetilde{g} \leftarrow ite(f_{\overline{v}}, g_{\overline{v}}, h_{\overline{v}})$;       do “terminal case” */
        if $(\widetilde{f}$ equals $\widetilde{g})$ return $\widetilde{g}$;
        $R \leftarrow$find_or_add_unique_table$(v, \widetilde{f}, \widetilde{g})$;   // postoji li novi čvor
        insert_computed_table$(\{f, g, h\}, R)$;       /* dodaj argumente i
        return $R$; } }                                rezultat R u izračunsku
                                                       tablicu i vrati R */
```

Normalizacija ITE-trojki i ekvivalencije

- Pojednostavljenja:

$$\begin{array}{rcl} ite(F, F, G) & \Longrightarrow & ite(F, 1, G) \\ ite(F, G, F) & \Longrightarrow & ite(F, G, 0) \\ ite(F, G, \overline{F}) & \Longrightarrow & ite(F, G, 1) \\ ite(F, \overline{F}, G) & \Longrightarrow & ite(F, 0, G) \end{array}$$

- Ekvivalencije:

$$\begin{array}{rcl} ite(F, 1, G) & \equiv & ite(G, 1, F) \\ ite(F, 0, G) & \equiv & ite(\overline{G}, 0, \overline{F}) \\ ite(F, G, 0) & \equiv & ite(G, F, 0) \\ ite(F, G, 1) & \equiv & ite(\overline{G}, \overline{F}, 1) \\ ite(F, G, \overline{G}) & \equiv & ite(G, F, \overline{F}) \end{array}$$

Koristiti lijevi zapis

↓

$$ite(F, G, H) \equiv ite(\overline{F}, H, G) \equiv \overline{ite(F, \overline{G}, \overline{H})}, \equiv \overline{ite(\overline{F}, \overline{H}, G)}$$

Ekvivalencije treba odabrati tako da prvi argument ima **raniju varijablu u redoslijedu uređenosti**

Složenost izračunavanja ITE(f, g, h)

- **Bez uporabe izračunske tablice**
  - Jedan pristup jedinstvenoj tablici (bez rekurzije) = konst. vrijeme.
  - Rekurzivni poziv (ako nije završni) traži 2 nova pristupa tablici.
  - Vrijeme izvođenja je **eksponencijalno** prema broju varijabli.
- **Uz uporabu izračunske tablice i s neograničenom memorijom**:
  - Neka je $|f|$ broj čvorova u ROBDD-u za f, (analogno za g i h).
  - Za jedinstvenu kombinaciju (f, g, h) funkcija ***ite(f, g, h)*** se poziva jednom.
  - Općenito ***ite(f, g, h)*** se poziva $|f|\cdot|g|\cdot|h|$ puta, pa je složenost:

    $O(|f|\cdot|g|\cdot|h|)$
  - Za ***ite*** funkciju s dva operanda (npr. logičke funkcije AND, OR, XOR i sl.), složenost je $O(|f|\cdot|g|)$ u najgorem slučaju.
- **Uz uporabu izračunske tablice i ograničenom memorijom**
  - **U najgorem slučaju eksponencijalno** prema broju varijabli
  - **U realnim aplikacijama bliže** $O(|f|\cdot|g|\cdot|h|)$

ROBDD i ITE – efikasnost uporabe

- Za izgrađeni ROBDD neke logičke funkcije, sljedeće se operacije mogu izvesti u konstantnom vremenu (vrlo teško za oblik SOP ili POS):
- Test na tautologiju i (ne)zadovoljivost: - samo jedan završni čvor

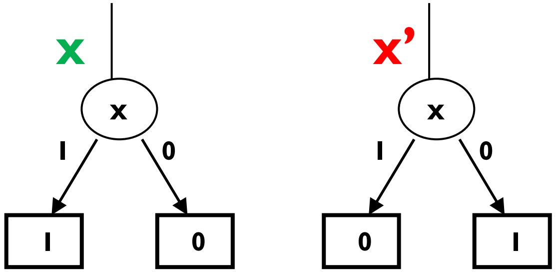
***f* = 1 ?** ***ite*(*f*, 1, 0)**

***f* = 0 ?** ***ite*(*f*, 0, 1)**

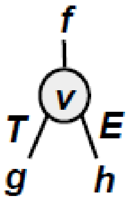
- Test na **ekvivalenciju dva ROBDD-a** provjerava za svaki čvor da su pokazivači na THEN i ELSE čvorove isti za obje funkcije – linearno ovisno samo o broju čvorova.

ROBDD – proširenje oznakama komplementa

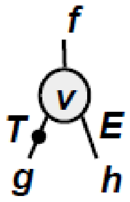
- BDD-ovi za neke dvije funkcije G i G’ su vrlo slični, jedina razlika su vrijednosti čvorova “djece”, koji su međusobno zamijenjeni



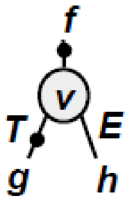
- Kako bi se iskoristio postojeći izgrađeni ROBDD za funkciju G, uvodi se komplementirani luk (pomoću punog kružića na luku) za funkciju G’, čije je značenje da se **invertira rezultat** (podfunkcija) na koji pokazuje luk. To se provodi radi **optimizacije prostora i vremena pretraživanja**



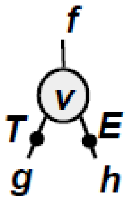
$f = vg+v'h$



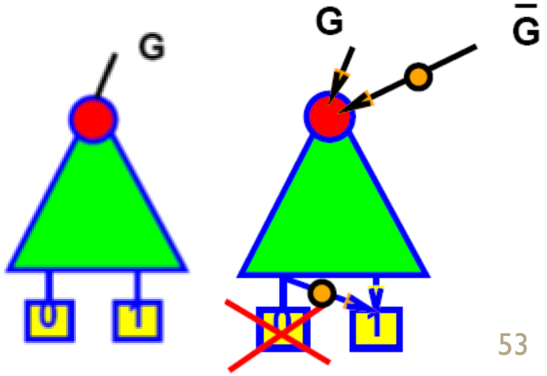
$f = vg'+v'h$



$f' = vg'+v'h$

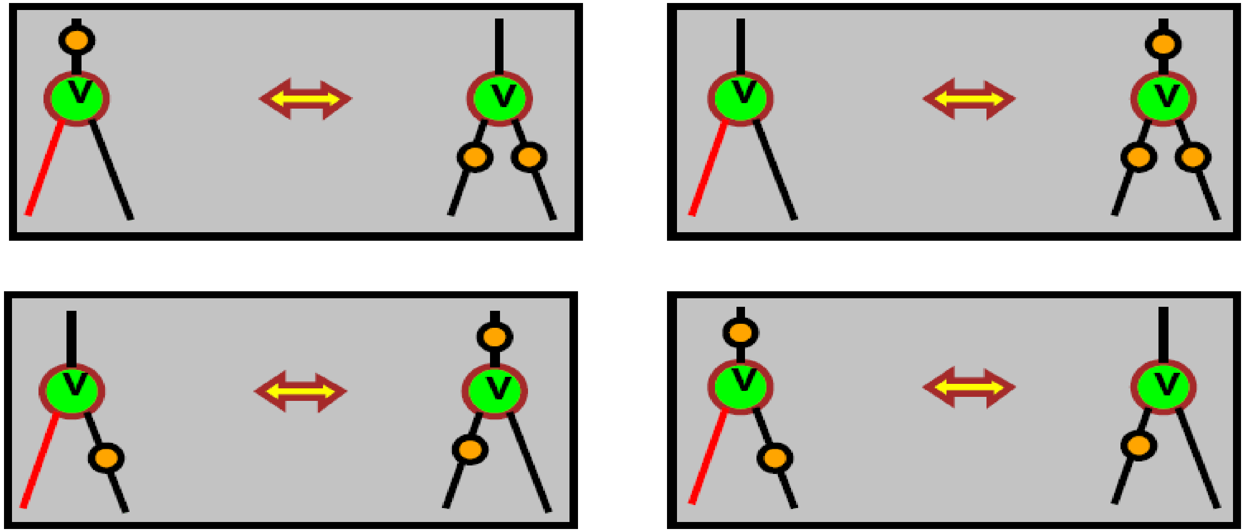


$f = vg'+v'h'$



ROBDD – proširenje oznakama komplementa

- Uvođenje komplementiranih lukova **narušava kanoničnost prikaza**, jer postoje logički ekvivalentni oblici funkcija. Ta nekanoničnost se mora razriješiti:



- Rješenje (konsenzus): **odaberi onaj oblik za koji komplementirani luk nije na THEN strani**

ROBDD – proširenje oznakama komplementa – postupak izgradnje

- Uvođenje komplementiranih lukova za neku funkciju *f*:
  1. Nacrta se uobičajeni ROBDD
  2. Zadrži se samo završni čvor 1, prijelazi koji idu u 0 se komplementiraju i prebace u 1
  3. Iterira se od razine zadnjih nezavršnih čvorova prema korijenu:
     1. Primijeni se ekvivalentni kanonični prikaz (vidi prethodni slajd)
     2. Ako postoje čvorovi s THEN stranom istom kao ELSE stranom, uklone se
     3. Riješe se izomorfni podgrafovi, ako postoje (ostavi se samo jedan)
  4. Kada se na taj način izgradila funkcija *f*, može se uvijek dodati u korijenu komplement za funkciju *f* ’

Kao rezultat ovog postupka, ako se za neku interpretaciju varijabli prođe BDD-om kroz **paran** broj **komplementiranih** lukova, rezultat je **1**, a ako se prođe kroz **neparan** broj, rezultat je **0**.

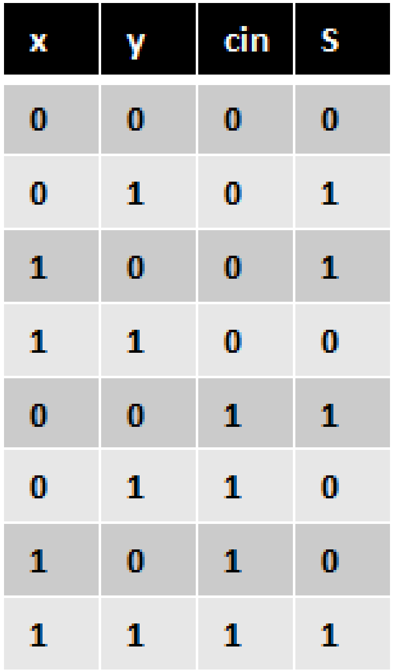
Zadaci

3. Za Booleovu karakterističnu funkciju iz zadatka 1 nacrtajte ROBDD uz proizvoljno uređenje varijabli.

4. Za funkciju **S** sume potpunog zbrajala zadanu tablicom 1, nacrtajte ROBDD te provedite komplementiranje lukova uz uređenje x < y < cin.

5. Za funkciju **F = acd + bc + a'd'** izgradite ROBDD primjenom ITE- algoritma (rekurzivni postupak, uz potrebna pojednostavljenja) i uz uređenje a < d < c < b.
   Napomena: potrebno je raspisati cjelokupni rekurzivni postupak i nacrtati konačni ROBDD.

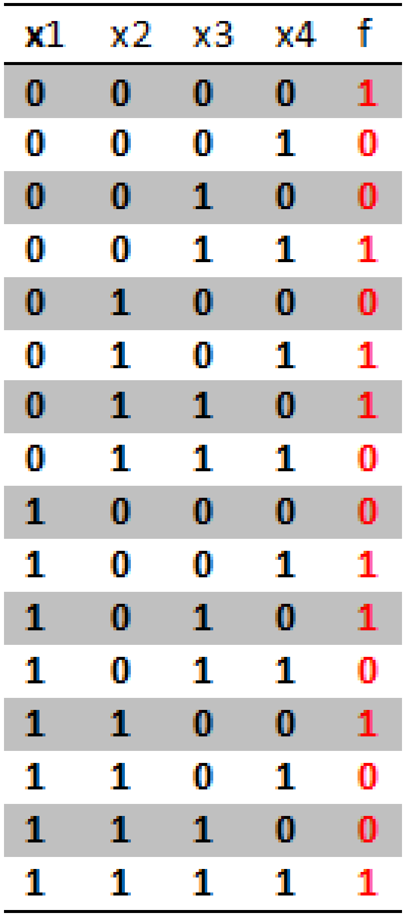
Tablica 1



x	y	cin	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	1

Zadaci

6. Odredite složenost prikaza BDD-om funkcije parnog pariteta od *n*-varijabli (dan je tablični prikaz nadesno za 4 varijable), neovisno o uređenju



x1	x2	x3	x4	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

PRIMJENA BDD-OVA

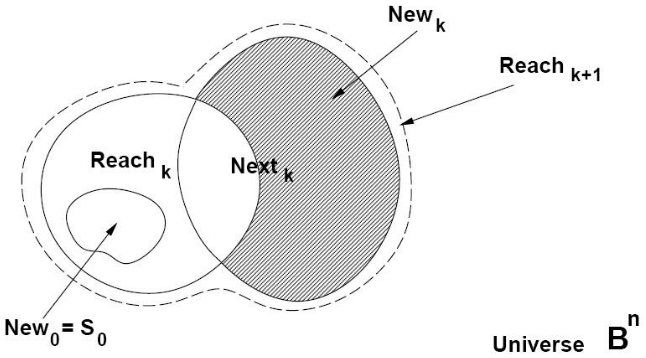
Simbolička analiza dosezljivosti BDD-ovima

- **Izračunavanje sa skupovima preslikava se u izračunavanje ROBDD-ovima**
- Skupovi ***S*** i ***T*** dani su karakterističnim funkcijama (kodiranjem) χ_S i χ_T , a one s ROBDD-ovima
- ROBDD-ovi se mogu implementirati ITE-algoritmom

Set	BDD
$\varnothing$	`BDD_0`
$\overline{S}$	`bdd_not(S)`
$S \cup T$	`bdd_or(S,T)`
$S \cap T$	`bdd_and(S,T)`
$S = T$	`S=T`
Universe	`BDD_1`

***ite* izračun**
0
ite(S, 0, 1)
ite(S, 1, T)
ite(S, T, 0)
ite(S, T, T')
1

Simbolička analiza dosezljivosti



```
$2^S$ explore ($S_0$,$H$)
{
  k:=1;  Reach:=$\emptyset$;
  do {
    New_Reach:=$S_0 \cup H$(Reach);
    if (New_Reach = Reach)
      return Reach;
    k++;
    Reach:=New_Reach;
  } forever;
}
```

```
BDD explore (BDD $S_0$, BDD (*$H$)(BDD))
{
  Reach:=BDD_0;
  do {
    New_Reach:=bdd_or($S_0$,$H$(Reach));
    if (New_Reach=Reach)
      return Reach;
    Reach:=New_Reach;
  } forever;
}
```

Simbolička provjera modela ROBDD-om

- Razmatramo provjeru modela za specifikaciju zadanom **CTL formulom i koristimo teoriju fiksne točke**

1. **Izračun formule: EX p**
   - Logička formula ***p*** dana je svojim ROBDD-om (skupom stanja u kojima ***p* = *true***).

```
BDD EX (BDD p)
{
    return H^-1(p);
}
```

$H(p) = \{t \in S \mid \exists s \in (p \cap R)\}$

```
BDD H(BDD p, BDD R) {
return bdd_exist(s, bdd_and(p, R))}
```

s je vektor svih stanja u zadanom početnom skupu.
Disjunkcija kofaktora, isključuje početna stanja **s:**
`bdd_exist(s, f):` $\exists_s f = f_s + f_{s'} = ite(f_s, 1, f_{s'})$

   - Kako je $H: \{s\} \rightarrow \{t\}$, to je $H^{-1}: \{t\} \rightarrow \{s\}$.
   - Operacija se u ROBDD-u postiže jednostavnom zamjenom s_i sa t_i. To je moguće napraviti jednim prolazom kroz ROBDD ako je redoslijed varijabli: $s_1, t_1, s_2, t_2, \ldots$

Simbolička provjera modela ROBDD-om

2. Izračun formule CTL vremenske logike: EG p

- Logička formula ***p*** dana je svojim ROBDD-om

  F (Z) = Z

- Izračun skupovima: Q(EG p) = Q(p) $\cap$ Q(EX EG p)
- Izračun ROBDD-ovima:

```
BDD EG (BDD p) {
    k:=1;  $Z_k$:=p;
        do  {
            $Z_{k+1}$:= bdd_and(p, $H^{-1}(Z_k)$);
            if ($Z_{k+1}$ = $Z_k$) return $Z_k$ ;
            k++;
        } forever; }
```

Simbolička provjera modela ROBDD-om

3. Izračun formule CTL vremenske logike:

E(p U q) = (p EU q)

  - Logičke formule ***p*** i ***q*** dane su svojim ROBDD-ovima.

F (Z) = Z

- Izračun skupovima: Q(p EU q)) = Q(q) ∪ Q(p) ∩ Q(EX (p EU q))
- Izračun ROBDD-ovima:

```
BDD EU (BDD p, BDD q) {
    k:=1;  $Z_k$:=q;
      do  {
          $Z_{k+1}$:= bdd_or(q, bdd_and(p, $H^{-1}(Z_k)$));
          if ($Z_{k+1}$ = $Z_k$) return $Z_k$ ;
          k++;
          } forever; }
```

Prisjetimo se ekvivalencija...

- $AX\ \varphi = \neg EX\ \neg\varphi$
- $AF\ \varphi = \neg EG\ \neg\varphi$
- $AG\ \varphi = \neg EF\ \neg\varphi = \neg E\ (True\ U\ \neg\varphi)$

Dodatno:

- $A\ (\psi\ U\ \varphi) = \neg E\ (\neg\varphi\ U\ \neg\psi \wedge \neg\varphi) \wedge AF\ \varphi$

- Stoga je moguće sve ostale vremenske formule izračunati uz pomoć postojeća tri algoritma EX, EG, EU koja računaju s BDD-ovima

Kako to ide s BDD-ovima u NuSMV-u

1. Pokrene se NuSMV u interaktivnom načinu

   > NuSMV -int

2. Učita se datoteka s modelom

   > read_model –i file.smv

3. Sravni se hijerarhija modula, čime se instanciraju moduli i procesi

   > flatten_hierarchy

4. **Varijable modula i procesa se kodiraju**, čime se dobivaju njihove karakteristične funkcije

   > encode_variables (po defaultu, stvaraju se prolaskom po dubini modela)

5. Za kodirane varijable gradi se ukupni BDD slijedeći odgovarajuće relacije prijelaza i početna stanja

   > build_model

6. Uz pomoć izgrađenog BDD-a provjerava se doseziljivost stanja, provjera specifikacije i drugo

   > compute_reachable , check_fsm , check_ctlspec...

Redoslijed varijabli u NuSMV-u

- Po *defaultu*, prilikom poziva naredbe `encode_variables` redoslijed varijabli je određen prolaskom u dubinu kroz hijerarhiju modela
- Može se navesti ulazna datoteka s eksplicitno zadanim poretkom varijabli `encode_variables -i input_order_file`
- Moguće je uvesti heuristiku *basic* kojom se grupiraju varijable koje su zajedno u prijelazima Kripke strukture (`bdd_static_order_heuristics`)
- Dinamičko preuređivanje događa se ako broj čvorova dostigne neki prag ili ako korisnik to eksplicitno pokrene
- Naredba za pokretanje dinamičkog preuređivanja je `dynamic_var_ordering -e <method>`, gdje je `<method>` neka od metoda iz skupa:
  - *Sift* – pomiče jednu po jednu varijablu kroz poredak dok ne nađe najbolji poredak
  - *Random* – parovi slučajno odabranih varijabli zamijene mjesta
  - *Window* – permutira varijable unutar prozora od 2,3 ili 4 susjedne varijable
  - Ostalo (*genetic*, *annealing*...)

Zaključak

- ROBDD-ovi su zanimljive i uporabive strukture podataka.
- Zahvaljujući ROBDD strukturama, mnogi verifikacijski problemi mogu se uspješno riješiti.
- *"There is no such thing as a free lunch"*: Postoje arhitekture sklopova (npr. sklop za množenje), te pojedinih programskih dijelova koje se ni sa ROBDD-ovima ne mogu riješiti u polinomnom vremenu i prostoru
- Najpoznatiji programski paketi:
    - **CMU BDDLIB**: Carnegie Mellon University, Pittsburgh.
    - **CUDD**: BDD package, University of Colorado, Boulder.
- ROBDD-ovi su uključeni u većinu suvremenih alata za provjeru modela i provjeru ekvivalencije

Zadatak za bonus bod

- Nacrtajte ROBDD dijagram pojednostavljen s komplementiranim lukovima za logičku funkciju:

$$F(x1,x2,x3) = (x1 + x2)(x1 + x3)(x2 + x3)(x1' + x2')$$

uz uređenost varijabli $x3 < x1 < x2$.